Volker Roth

Rechnersicherheit, SoSe 21

Übung 08

TutorIn: Oliver Wiese
Tutorium 02
Materialien: Latex, VSC, Skript

17. Juni 2021

---

1 Honeywords

Imagine you are advising the administrators of three websites (1.example.org, 2.example.org and 3.example.org) on the use of honeywords for their systems. The three web sites are run by different companies and managed by distinct administrators. You want to explain the configuration of honeywords and it's benefits using an example of two users, Alice and Bob, both having accounts on all three web sites. Alice wants to use three separate passwords for 1, 2 and 3, while Bob wants to use a single password for all web sites.

Remark 1: You must assume that the process how decoys are generated is publicly known. **Remark 2**: The goal of this exercise is the thought-experiment of implementing honeywords in practice. You are allowed to make your own assumptions where necessary and realistic, and are encouraged to use common sense where necessary. In particular, watch out for difficult or undesirable situations (e.g. if your solution would require that all websites have to share the same password database).

(a) *Suggest how websites 1 and 2 should implement honeywords to protect Alice and Bob from password cracking. Explain your decisions. (If you need to, you may make assumptions on how Alice's and Bob's passwords look like on 1 and 2 - for example enforced by the password registration process.)*

1. *How should decoy passwords be constructed?*

   - The following points are significant
     - The honeywords should be believable (not just random chars), else they are easily detectable after cracking.
     - They should not be to hard to crack. Otherwise the adversary would simply skip the password.
     - They should not be easy to crack, otherwise it is too suspicious.
     - If possible, the user should not accidentally access the decoy account, e.g. if he enters his password incorrectly (not too similar passwords). Else this would result in false alarms.
     - The honeywords should not be too similar to the user password. Otherwise the adversary could possibly derive the correct password from the honeywords.

- We came up with three ideas for creating honeywords:
    1. Take the user's password and remove, change and or add chars. (e.g. added or changed chars can be random or depend on other users for data like his age, name, phone number)
    2. Check the password of the user and with this create your own passwords accordingly. E.g. by using a dictionary and some random chars.
    3. Check the password of the user and choose a similar password with a password model. E.g. users password and password model password have similar probability's.
- And we found three generating functions for creating honeywords in the decoy password paper:
    1. **Tweaking**
        - <u>Pro:</u>
            * Is easy to implement
        - <u>Con:</u>
            * Head of PW is alwasy exposed
            * Requires a random user password in order to create good honeywords from it
    2. **Chaffing**
        - <u>Pro:</u>
            * Honeywords and the real password are not distinguishable. (honeywords look realistic)
        - <u>Con:</u>
            * A good password model is needed
    3. **Random pick**
        - <u>Pro:</u>
            * Is easy to implement
            * Easy to use in combination with password managers
        - <u>Con:</u>
            * Irritating for user (as he has to pick multiple passwords)
            * User has to forget the honeywords! There is a high chance that he will log into an decoy account

  We decided to 'use' Chaffing, as this way the honeywords are not too similar and not distinguishable of the user's password.

2. *How many decoy passwords should be generated?*
    - Many, so that the chance that the adversary chooses a honeyword is high.
    - But not too many, otherwise the adversary could possibly recognize a pattern in the honeywords and thus rule it out.

   In the paper N $\geq$ 10,000 is recommended for practical use.

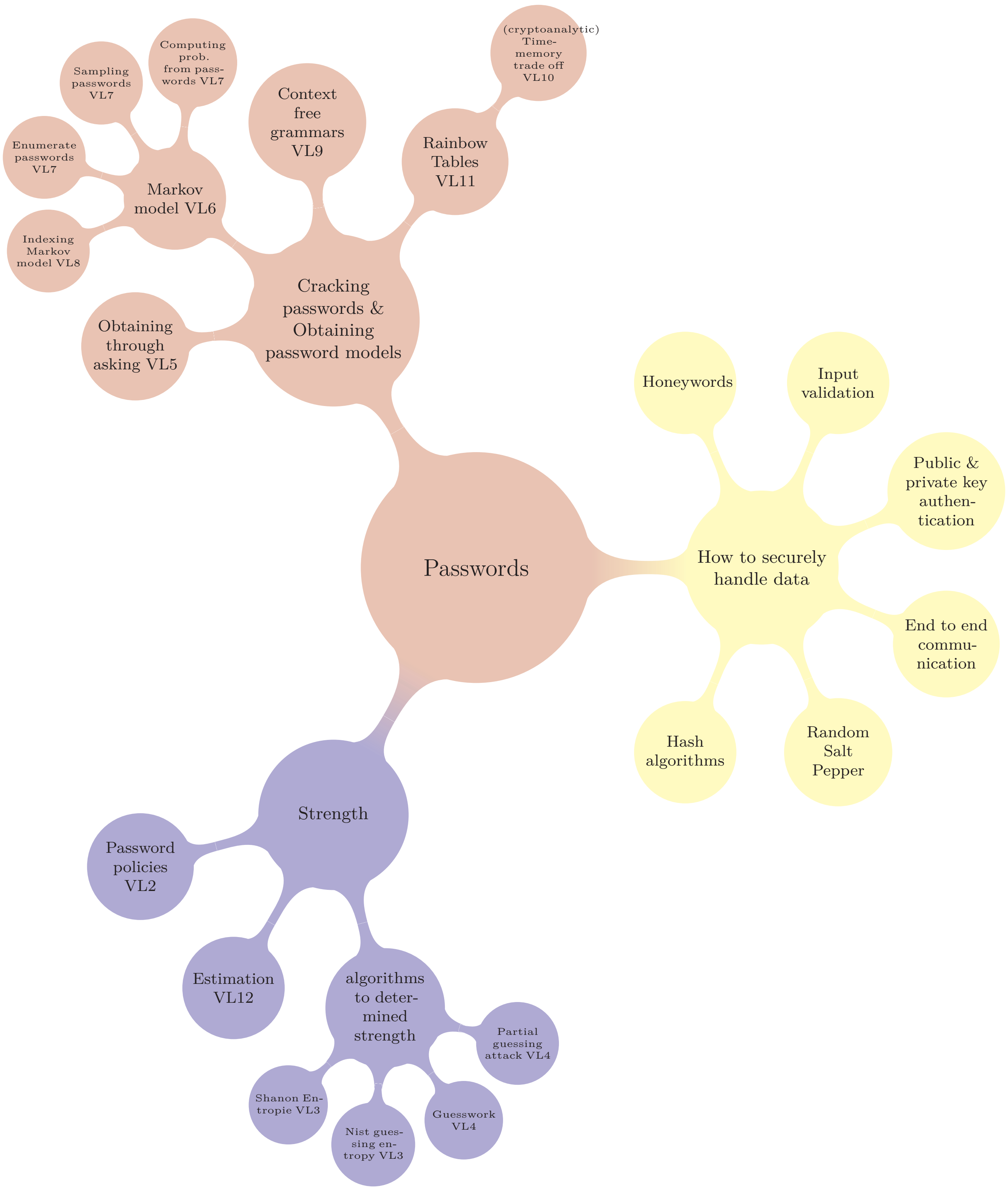
(b) *Using your construction, analyze the success probability of an adversary impersonating Alice and Bob to 3, who successfully breaches (excluding the honeychecker)*
- *only 1.example.org or 2.example.org*

  - We assume that the adversary cannot distinguish between honeywords and the real password.
  - This results that the adverse for Bob and Alice has a $\frac{1}{10{,}000}$ chance of guessing the correct password.

- *both.*
  - Since he has the password data from 1.example.org and 2.example.org, he could just compare them. Then he might notice that one password is exactly the same in both data sets, and this would indeed be the correct password. However, this does not affect Alice, as she has a different password for each page. $\rightarrow$ Alice $\frac{1}{10{,}000}$, Bob 1.
  - To get around this, you could use the same honeywords for users who use exactly the same login data on different pages.
  - This only works for pages over which we have control. Because the problem remains if Bob also has an account with the same login data on a page we do not own. Example: Bob has an account on 4.example.org that does not belong to us. 1.example.org and 4.example.org are affected by data leaks. Both sites use honeywords, but Bob used the same logins for both sites. If the adversary now compares the passwords, he would again notice the correct password.

2 Recap Passwords

(a) *We discussed different aspects of passwords. You should summarize and visualize them,e.g. drawing, painting,... (best case mind map)*



Passwords
Cracking passwords & Obtaining password models
Obtaining through asking VL5
Markov model VL6
Enumerate passwords VL7
Sampling passwords VL7
Computing prob. from passwords VL7
Indexing Markov model VL8
Context free grammars VL9
Rainbow Tables VL11
(cryptoanalytic) Time-memory trade off VL10
How to securely handle data
Honeywords
Input validation
Public & private key authentication
End to end communication
Random Salt Pepper
Hash algorithms
Strength
Password policies VL2
Estimation VL12
algorithms to determined strength
Partial guessing attack VL4
Guesswork VL4
Nist guessing entropy VL3
Shanon Entropie VL3

password123
user_account
Password_policy
hashed
random_salt_pepper
end_to_end_encrypted
input_validation
decoy_passwords
key_authentication